# Chapter 1

# Tools and Environment

Before we can program a Robot, we need to set up our programming environment. There are three aspects of the environment that need to be set up before we can begin: the Compiler, the Pros API, and Git version control. We also will set up our editor to mostly provide autocomplete and error checking, however most error checkers flag the API header, so that is of limited utilty. I suggest using VSCode (VSCodium is a Free and Open Source (FOSS, See GNU.org for more info) version) or Vim. VSCode is much easier to use and set up but Vim is often faster and more performant for writing code. If you are new to programming, I suggest sticking with VSCode, but if you are already experienced configuring Vim is a fun experience. Either way, instructions are included to set both up.

## 1.1 Configuring the Editor

### 1.1.1 VSCode (Recommended for New Programmers)

VSCode is a free and mostly open source editor from Microsoft, that usually requires limited configuration and has an intuitive extension system. VSCodium is almost exactly the same but does not include proprietary content, and is fully FOSS. Download links for VSCode on major systems can be found at `https://code.visualstudio.com/download`. After following the installation instructions, we need to install a few packages. The first package that should be installed is the Microsoft C/C++ extension. This will provide basic autocomplete and error checking. We should install GitLens as it is a handy extension for version control. Git can also be accessed from the command line, which is my preferred method, but GitLens is usually more intuitive. We can optionally install the PROS extension. It can be handy but it often causes a big performance hit. All instructions in this book are given as command line instructions, so this extension is not required nor is it recommended; only install this extension if you want.

To use VSCode with your project, open the root folder of your project. This can be done using `Crtl+K Crtl+O` or `code <directory>` from the command line. In VSCode, all the files can be found in the explorer and opened for editing. Also, VSCode has a built in terminal that can be accessed with `Crtl+‘` (the key below escape). This terminal is the primary way we will interact with the PROS API.

For more information on VSCode, there are numerous videos on the internet that concern more advanced usage. I recommended https://www.youtube.com/watch?v=ifTF3ags0XI to get a good handle on shortcuts for faster editing.

### 1.1.2 Vim (Not Recommended for the Inexperienced)

Vim is a powerful command line based editor, known for it’s odd but very efficient keyboard shortcuts. Vim can be easily installed through any Linux package manager. Download links for Windows and Mac can be found at `https://www.vim.org/download.php`. Before continuing, run the command `vimtutor` and work through the exercises. This will introduce you to the most important movements for Vim allow efficient use of the editor. (From this point on, all Vim related instructions are intended for Linux, other operating systems may or may not work.) If you still wish to continue with Vim, you should set up a `.vimrc` in your home directory a good resource can be found on https://www.freecodecamp.org/news/vimrc-configuration-guide-customize-your-vim-editor/.

You will also want to set up https://github.com/junegunn/vim-plug and https://github.com/VundleVim/Vundle.vim We will use vim-plug to install https://github.com/dense-analysis/ale, and we will use Vundle to install https://github.com/ycm-core/YouCompleteMe.

The current Github Readme for YCM says that Vundle is currently broken. This seems to be outdated as I was able to install it using Vundle. Following the existing instructions should work but can be difficult and may take a few hours to install correctly.

There is no PROS plugin or GitLens for Vim, so these are all the needed plugins.

For more information on Vim, see the website at `https://www.vim.org/`.

## 1.2 Introduction to PROS API

To interface with the VEX Brain, we will use the Purdue PROS API. PROS is a high performance C/C++ API from Purdue university that is accessed through the command line. Although the VSCode extension exists, and you may have installed it, all instructions related to the API will be given as command line commands.

### 1.2.1 Setup

First we will install a cross compiler to build for the Vex Brain. After that we ill install the PROS toolchain using the Python package manager `pip3`. Finally we will test our environment to ensure the the API is working correctly. The instructions provided should work for Windows, Linux, and macOS; with minor differences between them. If you use WSL, there is an extra step to allow uploads to the robot brain.

For more information see the https://pros.cs.purdue.edu/v5/getting-started/. We will be following the Linux instructions found here: `https://pros.cs.purdue.edu/v5/getting-started/linux.html`, they are similar but not exactly the same for Windows and macOS.

#### Installing the Crosscompiler

Since the Brain runs a different OS and the processor has a difficult architecture, we require a compiler that will build for the ARM architecture. For this purpose, we will use the ARM GNU Toolchain which can be downloaded https://developer.arm.com/downloads/-/gnu-rm. You need to decompress the file, and add the `bin` directory to your system PATH. (These are called environment variables on Windows). On macOS and Linux, you can add export `PATH=$PATH:/path-of-toolchain/bin` to your `.bashrc`[1] (replace `path-to-toolchain` with the path of the decompressed directory) Then run `source ~/.bashrc` or open a new shell to load changes.

To check if the installation was successful, run `arm-none-eabi-gcc --version` from outside the toolchain's `bin` directory. If it prints out the and copyright information: Success! If it signals a command not found error, check that the path has been configured properly and that the changes have been loaded.

[1] or the equivalent of your shell